UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/885,705	06/20/2001	Akram Boughannam	6169-243	4240

40987 7590 12/09/2005

AKERMAN SENTERFITT
P. O. BOX 3188
WEST PALM BEACH, FL 33402-3188

EXAMINER	
PAULA, CESAR B	

ART UNIT	PAPER NUMBER
2178	

DATE MAILED: 12/09/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/885,705	BOUGHANNAM, AKRAM	
	Examiner	Art Unit	
	CESAR B. PAULA	2178	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *13 October 2005*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-21* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-21* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the amendment filed on 10/13/2005.

This action is made Final.

2. In the amendment, claims 1-21 are pending in the case. Claims 1, 7, 11, 14, and 16 are independent claims.

Drawings

3. The drawings filed on 6/20/2001 have been approved by the examiner.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1-6, 11, and 16-21 remain rejected under 35 U.S.C. 103(a) as being unpatentable over Anthony, in view of Fowler et al, hereinafter Fowler, "UML Distilled: A Brief Guide to the Standard Object Modeling Language" 2nd ed., Addison Wesley, 8/1999, chapt.8, and further in view of Daugherty et al, hereinafter Daugherty (US Pub. #2002/0016828 A1, 2/7/2002, filed on 12/3/1998).

Regarding independent claim 1, Anthony teaches translating -- *loading* --a semantic representation of a first document written in an XML format to a second XML format (0353, 0366). Anthony fails to explicitly disclose *loading state chart data corresponding to a state chart diagram, and specifying life-cycle states possible for each object and behavior exhibited by said objects for each specified state, said state chart data being specified according to a modeling language.* Fowler teaches generating a state diagram indicating the behavior of two objects or items in an order processing system. The state diagram, in UML-- *a modeling language*--, shows several states—*life cycle states*-- of the items in the system, and the behavior, such as check item, initiate delivery, etc., for such items for each displayed state—*specifying life-cycle states possible for each object and behavior exhibited by said objects for each specified state* --(page 2, fig.8-1). It would have been obvious to one of ordinary skill in the art at the time of the invention to combine Anthony, and Fowler, to generate the semantic representation of the first XML document using the flexibility of UML, because Anthony teaches the translation of XML documents using a first dtd into XML documents using second dtds(0009, 0011) . This provides the benefit of smoothly, and efficiently exchanging information among users who implement different XML document formats.

Further, Anthony fails to explicitly disclose *state transition data specifying event occurrences for transitioning from said state to another specified state.* Fowler teaches generating state diagram(s) having state names, and transition data, such as "item received[all items available]", for specifying events for transitioning to another state (in this case "Dispatching")— *state transition data specifying event occurrences for transitioning from said state to another specified state* --(pages 2-3, fig.8-1). It would have been obvious to one of

ordinary skill in the art at the time of the invention to combine Anthony, and Fowler, because Anthony teaches the translation of XML documents using a first dtd into XML documents using second dtds(0009, 0011) . This provides the benefit of smoothly, and efficiently exchanging information among users who implement different XML document formats.

Further, Anthony teaches translating a first document written in an XML format to a second XML format, such as document in fig.19, which has a tag describing the version—"<?xml version="1.0?>" (0353-0356, 0366).Anthony fails to explicitly disclose *generating header and footer data in accordance with a selected markup.* However, Daugherty teaches an XML file describing a provider header, and footer (0045). It would have been obvious to one of ordinary skill in the art at the time of the invention to combine Anthony, Fowler, and Daugherty, because Daugherty teaches above XML descriptions for maximizing the cacheability of a web page stock data. Thus, enabling a user to quickly retrieve the web page.

Moreover, Anthony teaches the translation —*retrieval* to translate--by a system of a model (0365-0367, fig.20). Anthony fails to explicitly disclose --*state name and state transition data.* Fowler teaches generating a state diagram having state names, and transition data, such as "item received[all items available]", for specifying events for transitioning to another state (in this case "Dispatching") (pages 2-3, fig.8-1). It would have been obvious to one of ordinary skill in the art at the time of the invention to combine Anthony, and Fowler, because Anthony teaches the translation of XML documents using a first dtd into XML documents using second dtds(0009, 0011) . This provides the benefit of smoothly, and efficiently exchanging information among users who implement different XML document formats.

Furthermore, Anthony teaches the translation of a first XML document model into a second XML document—*formatting and saving retrieved state name and transition according to* XML (0366-0367, fig.20).

Regarding claim 2, which depends on claim 1, Anthony teaches the translation of a first XML document into a catalog model. The translation into a second XML document (0366-0367, fig.20). Anthony fails to explicitly teach *parsing said composite state action into individual state actions* . However, Fowler teaches the generating a state diagram of a system by including states, such as "Checking and Dispatching" states. The states are entered or performed sequentially, the "check" state first, and so on—*parsing said composite state action into individual state actions* (pages 2-3, and 5). It would have been obvious to one of ordinary skill in the art at the time of the invention to combine Anthony, and Fowler, because Anthony teaches the translation of XML documents using a first dtd into XML documents using second dtds (0009, 0011) . This provides the benefit of smoothly, and efficiently exchanging information among users who implement different XML document formats.

Regarding claim 3, which depends on claim 1, Anthony teaches the translation of a first XML document model into a second XML document (0366-0367, fig.20).

Regarding claim 4, which depends on claim 1, Anthony discloses generating a model of a first XML document (0365-0367, and fig.20). Anthony fails to explicitly disclose *said state chart diagram is a unified modeling (UML) state chart.* However, Fowler teaches generating a

UML state diagram having state names, and transition data, such as "item received[all items available]", for specifying events for transitioning to another state (in this case "Dispatching") (pages 2-3, fig.8-1). It would have been obvious to one of ordinary skill in the art at the time of the invention to combine Anthony, and Fowler, because Anthony teaches the translation of XML documents using a first dtd into XML documents using second dtds (0009, 0011). This provides the benefit of smoothly, and efficiently exchanging information among users who implement different XML document formats.

Regarding claim 5, which depends on claim 3, Anthony teaches a DTD defining structure of an XML document (0355-0363).

Regarding claim 6, which depends on claim 5, Anthony teaches the translation — *retrieval* to translate--by a system of a model (0365-0367, fig.20). Anthony fails to explicitly disclose --*state name and state transition data.* Fowler teaches a state diagram having state names, and transition data, such as "item received[all items available]", for specifying events for transitioning to another state (in this case "Dispatching") (pages 2-3, fig.8-1). It would have been obvious to one of ordinary skill in the art at the time of the invention to combine Anthony, and Fowler, because Anthony teaches the translation of XML documents using a first dtd into XML documents using second dtds (0009, 0011) . This provides the benefit of smoothly, and efficiently exchanging information among users who implement different XML document formats.

Regarding independent claim 11, Anthony discloses translating a first XML document into a catalog model using a translator—*state machine modeling tool*-- (0366-0367). Anthony fails to explicitly disclose *state chart names, and transition data, composite state actions, state chart data specified according to a modeling language.* Fowler teaches the generating a state diagram, in UML-- *a modeling language*--, of an order processing system by including states, such as "Checking and Dispatching"—*names*-- states, lines representing the transition from one state to the next, and actions, such as "do/check, and do/dispatch". The states are entered or performed sequentially, the "check" state first, and so on (pages 2-3, fig.8-3). It would have been obvious to one of ordinary skill in the art at the time of the invention to combine Anthony, and Fowler, to generate the semantic representation of the first XML document using the flexibility of UML, because Anthony teaches the translation of XML documents using a first dtd into XML documents using second dtds (0009, 0011) . This provides the benefit of smoothly, and efficiently exchanging information among users who implement different XML document formats.

Moreover, Anthony discloses translating the model into a second XML document (0365-0367, fig. 20). Thereby, parsing and translating classes, instances, and arrows into the second XML document—*parsing state actions into component state actions, formatting said state chart data and component state actions according to selected markup language.* Anthony fails to explicitly disclose *specifies life-cycle states possible for each object and behavior exhibited by said objects for each specified state, and wherein said state transition data specifying event occurrences for transitioning from said state to another specified state.* Fowler teaches generating a state diagram indicating the behavior of two objects or items in an order processing

system. The state diagram shows several states—*life cycle states*-- of the items in the system, and the behavior, such as check item, initiate delivery, etc., for such items for each displayed state—*specifying life-cycle states possible for each object and behavior exhibited by said objects for each specified state.* The state diagram specifies transition data for transitioning from one of the states to the other, based on some event, such as having all items checked and some items not in stock (pages 2-3, and fig.8-3). It would have been obvious to one of ordinary skill in the art at the time of the invention to combine Anthony, and Fowler, because Anthony teaches the translation of XML documents using a first dtd into XML documents using second dtds (0009, 0011) . This provides the benefit of smoothly, and efficiently exchanging information among users who implement different XML document formats.

Claims 16-18, and 19-21 are directed towards a program for performing the steps found in claims 1-3, and 4-6 respectively, and therefore are similarly rejected.

6. Claims 7-8, and 12-15 remain rejected under 35 U.S.C. 103(a) as being unpatentable over Anthony, in view of Fowler, and further in view of "Laura Lemay's Web Workshop JavaScript", Lemay et al, hereinafter Javascript, Sams, 1996, pp.7-9.

Regarding independent claim 7, Anthony discloses generating a model of a first XML document using a translator for translating a semantic representation of a first document written in an XML format to a second XML format—*formatting state chart data according to a selected*

markup language (0365-0367, and fig.20). Anthony fails to explicitly disclose *generating state chart data, specifying life-cycle states possible for each object and behavior exhibited by said objects for each specified state, and an add-in script for formatting said state chart data, said state chart data being specified according to a modeling language.* Fowler teaches generating a state diagram indicating the behavior of two objects or items in an order processing system. The state diagram, in UML-- *a modeling language--*, shows several states—*life cycle states--* of the items in the system, and the behavior, such as check item, initiate delivery, etc., for such items for each displayed state— *generating state chart data, and specifying life-cycle states possible for each object and behavior exhibited by said objects for each specified state* --(page 2, fig.8-1). Javascript teaches the combining a Javascript program with HTML for performing certain functions (page 6, lines 9-page 7, line 3). It would have been obvious to one of ordinary skill in the art at the time of the invention to combine Anthony, Fowler, and Javascript, to generate the semantic representation of the first XML document using the flexibility of UML, and convert this representation into the target XML using a programming language such as Javascript, because Anthony teaches the translation of XML documents using a first dtd into XML documents using second dtds (0009, 0011). This provides the benefit of smoothly, and efficiently exchanging information among users who implement different XML document formats. Javascript discloses using a relaxed program environment, which makes it easier on the programmer writing the program (page 7, lines 4-17).

Regarding claim 8, which depends on claim 7, Anthony teaches the translation of a first XML document model into a second XML document (0366-0367, fig.20).

Regarding claim 9, which depends on claim 8, Anthony discloses generating a model of a first XML document (0365-0367, and fig.20). Anthony fails to explicitly disclose *said generated state chart diagram is a unified modeling (UML) state chart data.* However, Fowler teaches generating a UML state diagram having state names, and transition data, such as "item received[all items available]", for specifying events for transitioning to another state (in this case "Dispatching") (pages 2-3, fig.8-1). It would have been obvious to one of ordinary skill in the art at the time of the invention to combine Anthony, and Fowler, because Anthony teaches the translation of XML documents using a first dtd into XML documents using second dtds (0009, 0011). This provides the benefit of smoothly, and efficiently exchanging information among users who implement different XML document formats.

Regarding claim 10, which depends on claim 8, Anthony teaches a DTD defining structure of the translated or formatted XML document (0355-0363).

Regarding claim 12, which depends on claim 11, Anthony discloses generating a model of a first XML document using a modeling system (0365-0367, and fig.20). Anthony fails to explicitly disclose *said generated state chart diagram is a unified modeling (UML) specified state chart diagram.* Anthony fails to explicitly disclose *said state chart diagram is a unified modeling (UML) state chart.* However, Fowler teaches generating a UML state diagram having state names, and transition data, such as "item received[all items available]", for specifying events for transitioning to another state (in this case "Dispatching") (pages 2-3, fig.8-1). It would have been obvious to one of ordinary skill in the art at the time of the invention to

combine Anthony, and Fowler, because Anthony teaches the translation of XML documents using a first dtd into XML documents using second dtds (0009, 0011). This provides the benefit of smoothly, and efficiently exchanging information among users who implement different XML document formats.

Regarding claim 13, which depends on claim 12, Anthony teaches the translation of a first XML document model into a second XML document (0366-0367, fig.20).

Regarding independent claim 14, Anthony discloses generating a model of a first XML document using a modeling system (0365-0367, and fig.20). Anthony fails to explicitly disclose *said modeling tool producing a unified modeling (UML) specified state chart diagrams.* Anthony discloses generating a model of a first XML document (0365-0367, and fig.20). However, Fowler teaches generating a UML state diagram having state names, and transition data, such as "item received[all items available]", for specifying events for transitioning to another state (in this case "Dispatching") (pages 1-3, fig.8-1). It would have been obvious to one of ordinary skill in the art at the time of the invention to combine Anthony, and Fowler, because Anthony teaches the translation of XML documents using a first dtd into XML documents using second dtds (0009, 0011). This provides the benefit of smoothly, and efficiently exchanging information among users who implement different XML document formats.

Moreover, Anthony fails to explicitly disclose *an add-in script defining markup representations of said UML specified state chart diagrams.* However, Javascript teaches the combining a Javascript program with HTML for performing certain functions (page 6, lines 9-

page 7, line 3). It would have been obvious to one of ordinary skill in the art at the time of the invention to combine Anthony, and Javascript, and convert the semantic representation into the target XML using a programming language such as Javascript, because Javascript discloses using a relaxed program environment, which makes it easier on the programmer writing the program (page 7, lines 4-17).

Furthermore, Anthony fails to explicitly disclose *a state machine run-time engine executing said markup language representations.* However, Javascript teaches using browser for including Javascript objects in an HTML web page (page 6-page 7, lines 3, 18-33). It would have been obvious to one of ordinary skill in the art at the time of the invention to combine Anthony, and Javascript, because Javascript discloses using a relaxed program environment, which makes it easier on the programmer writing the program (page 7, lines 4-17).

Regarding claim 15, which depends on claim 14, Anthony discloses generating a model of a first XML document into a second XML document (0365-0367, and fig.20). Anthony fails to explicitly disclose *said markup language representations are XML representation of said UML specified state chart diagrams.* Anthony discloses generating a model of a first XML document (0365-0367, and fig.20). Anthony fails to explicitly disclose *said state chart diagram is a unified modeling (UML) state chart.* However, Fowler teaches generating a UML state diagram having state names, and transition data, such as "item received[all items available]", for specifying events for transitioning to another state (in this case "Dispatching") (pages 2-3, fig.8-1). It would have been obvious to one of ordinary skill in the art at the time of the invention to combine Anthony, Fowler, and Javascript, because Anthony teaches the translation of XML

documents using a first dtd into XML documents using second dtds (0009, 0011). This provides the benefit of smoothly, and efficiently exchanging information among users who implement different XML document formats.

Response to Arguments

Applicant's arguments filed 10/13/2005 have been fully considered but they are not persuasive. Regarding independent claims 1, 11, and 16, the Applicant asserts that the prior art fails to teach or suggest how to transform state chart data of a modeling language into a markup language representation, because Anthony teaches converting from one XML format to another XML format, which is different from the concept of state chart diagram (pages 10-12). The Examiner disagrees, because although Anthony fails to explicitly teach the use of state chart data for performing the translation, Anthony does teach the transformation of the first XML format into a semantic representation or model of the XML, and the semantic representation to the target XML format (366). Fowler teaches generating a state diagram indicating the behavior of two objects or items in an order processing system. The state diagram, in UML-- *a modeling language*--, shows several states—*life cycle states*-- of the items in the system, and the behavior, such as check item, initiate delivery, etc., for such items for each displayed state—*specifying life-cycle states possible for each object and behavior exhibited by said objects for each specified state* --(page 2, fig.8-1). It would have been obvious to one of ordinary skill in the art at the time of the invention to combine Anthony, and Fowler, and generating the semantic representation of the first XML document using the flexibility of UML, because Anthony teaches the translation of XML documents using a first dtd into XML documents using second dtds (0009, 0011). This

provides the benefit of smoothly, and efficiently exchanging information among users who implement different XML document formats.

Regarding claims 7, and 14, the Applicant indicates that nothing suggests an add-in script for transforming state chart data into a markup language representation (page 12, parag.4). Anthony fails to explicitly *an add-in script for formatting said state chart data, said state chart data being specified according to a modeling language.* Fowler teaches generating a state diagram indicating the behavior of two objects or items in an order processing system. The state diagram, in UML-- *a modeling language--*, shows several states—*life cycle states--* of the items in the system, and the behavior, such as check item, initiate delivery, etc., for such items for each displayed state— *generating state chart data, and specifying life-cycle states possible for each object and behavior exhibited by said objects for each specified state* --(page 2, fig.8-1). Javascript teaches the combining a Javascript program with HTML for performing certain functions (page 6, lines 9-page 7, line 3). It would have been obvious to one of ordinary skill in the art at the time of the invention to combine Anthony, Fowler, and Javascript, to generate the semantic representation of the first XML document using the flexibility of UML, and convert this representation into the target XML using a programming language such as Javascript, because Anthony teaches the translation of XML documents using a first dtd into XML documents using second dtds (0009, 0011). This provides the benefit of smoothly, and efficiently exchanging information among users who implement different XML document formats. Javascript discloses using a relaxed program environment, which makes it easier on the programmer writing the program (page 7, lines 4-17).

Conclusion

7. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

I. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure. **Cover, R., RE: Mapping a UML model to a DTD or Schema, lists.xml.org/archives/xml-dev/200101/msg00927.html , 1/27/2001,** and **http://xml.coverpages.org/swiftML.html, "swiftML for Business Messages.", 2002.**

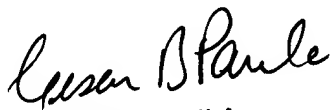
II. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Cesar B. Paula whose telephone number is (571) 272-4128. The examiner can normally be reached on Monday through Friday from 8:00 a.m. to 4:00 p.m. (EST).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Stephen Hong, can be reached on (571) 272-4124. However, in such a case, please allow at least one business day.

Information regarding the status of an application may be obtained from the Patent Application Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, go to http://portal.uspto.gov/external/portal/pair. Should you have any questions about access to the Private PAIR system, please contact the Electronic Business Center (EBC) at 866 217-9197 (toll-free).

Any response to this Action should be mailed to:
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450
Or faxed to:

- **(571)-273-8300** (for **all** Formal communications intended for entry)



CESAR PAULA
PRIMARY EXAMINER
12/6/05